Implement a C program to eliminate left factoring from a given CFG.

```c
#include <stdio.h>
#include <string.h>

#define MAX_RULES 10
#define MAX_LEN 50

char grammar[MAX_RULES][MAX_LEN];  // Stores the grammar rules
int ruleCount;                     // Number of rules

// Function to find the longest common prefix of two strings
void findCommonPrefix(char *s1, char *s2, char *prefix) {
    int i = 0;
    while (s1[i] == s2[i] && s1[i] != '\0' && s2[i] != '\0') {
        prefix[i] = s1[i];
        i++;
    }
    prefix[i] = '\0';
}

// Function to eliminate left factoring
void removeLeftFactoring() {
    printf("\nGrammar after eliminating left factoring:\n");

    for (int i = 0; i < ruleCount; i++) {
        char *rule = grammar[i];
        char nonTerminal = rule[0];  // Left-hand side (LHS)
        char rhs[MAX_RULES][MAX_LEN];
        int rhsCount = 0;

        // Split right-hand side (RHS) by '|'
        char *token = strtok(&rule[3], "|");
        while (token != NULL) {
            strcpy(rhs[rhsCount++], token);
            token = strtok(NULL, "|");
        }

        // Find the common prefix
        char prefix[MAX_LEN] = "";
        findCommonPrefix(rhs[0], rhs[1], prefix);

        if (strlen(prefix) > 0) {
            printf("%c -> %s%c'\n", nonTerminal, prefix, nonTerminal);
            printf("%c' -> ", nonTerminal);
            for (int j = 0; j < rhsCount; j++) {
                if (strncmp(rhs[j], prefix, strlen(prefix)) == 0) {
                    printf("%s", &rhs[j][strlen(prefix)]);  // Remove prefix
                } else {
                    printf("%s", rhs[j]);  // Print as is
                }
                if (j < rhsCount - 1) printf(" | ");
            }
            printf(" | ε\n");
        } else {
            printf("%s\n", grammar[i]); // No left factoring needed
        }
    }
}

// Main function
int main() {
    printf("Enter the number of grammar rules: ");
    scanf("%d", &ruleCount);

    getchar(); // Consume newline


    printf("Enter the grammar rules (format: A->α|β):\n");

    for (int i = 0; i < ruleCount; i++) {

        fgets(grammar[i], MAX_LEN, stdin);

        grammar[i][strcspn(grammar[i], "\n")] = '\0';  // Remove newline

    }


    removeLeftFactoring();

    return 0;

}
```

Input:

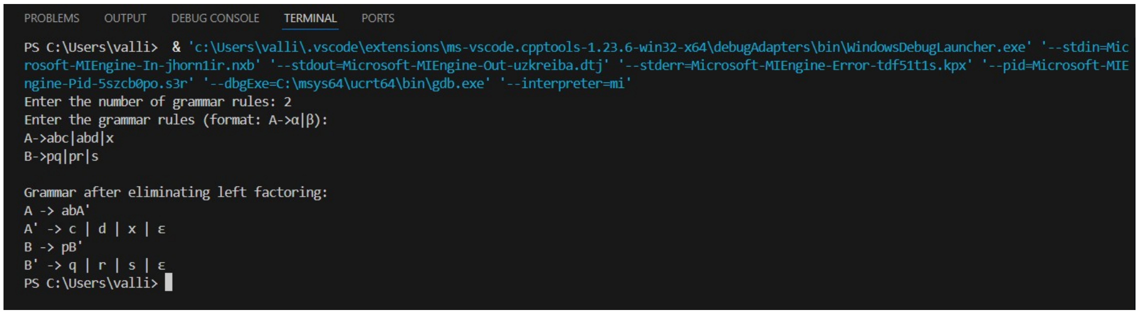
Enter the number of grammar rules: 2

Enter the grammar rules (format: A->α|β):

A->abc|abd|x

B->pq|pr|s

Output:



```
PS C:\Users\valli>  & 'c:\Users\valli\.vscode\extensions\ms-vscode.cpptools-1.23.6-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Mic
rosoft-MIEngine-In-jhorn1ir.nxb' '--stdout=Microsoft-MIEngine-Out-uzkreiba.dtj' '--stderr=Microsoft-MIEngine-Error-tdf51t1s.kpx' '--pid=Microsoft-MIE
ngine-Pid-5szcb0po.s3r' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'
Enter the number of grammar rules: 2
Enter the grammar rules (format: A->α|β):
A->abc|abd|x
B->pq|pr|s

Grammar after eliminating left factoring:
A -> abA'
A' -> c | d | x | ε
B -> pB'
B' -> q | r | s | ε
PS C:\Users\valli>
```